REMARKS/ARGUMENTS

The title of the invention has been amended.

Claims 1-26 are pending in the application. Claims 1-26 remain in the application. Claims 4, 5, 7, 12, 17 and 24-26 have been emended to remove quotation marks from certain terms in the claims.

Applicants respectfully request that the above-identified application be reconsidered in view of the following remarks:

Claim Rejections under 35 U.S.C.§ 103

Claims 1-6 and 14-26 are rejected under 35 U.S.C. §103(a) as being unpatentable over Kahle in view of McCrocklin in view of Shiell, U.S. Patent No. 5,864,697 (hereinafter "Shiell").

Kahle generally describes a method and system for data processing and, in particular, to a method and system for *emulating differing architectures* in a data processing system. Still more particularly, Kahle relates to a method and system for emulating guest branch instructions in a data processing system. (*See* Kahle, *Technical Field,* col. 1, lines 37-32). Applicants respectfully submit that Kahle's technical field or art differs from that of the present invention's.

McCrocklin describes a direct-execution microprogrammable microprocessor system using an emulatory microprogrammable microprocessor for direct execution of microinstructions in main memory through a microinstruction port. (See McCrocklin, *Abstract*).

Shiell discloses a microprocessor and system incorporating the same, utilizing combined actual branch history and speculative branch history to predict branches. (See Shiell, *Abstract*).

As per claim 1, Examiner states that Kahle has taught a method of detecting, recovering from and preventing bogus branch instructions in a microprocessor, the method comprising: predicting by branch prediction logic whether the at least one micro-op is a branch; executing the at least one micro-op; determining if the at least one executed micro-op is a bogus branch of the first macro instruction; and continuing processing with a second macro instruction, wherein if the at least one executed micro-op is determined to be branch, then the method comprises: flagging any other micro-ops which pertain to the at least one executed bogus branch micro-op; removing the flagged micro-ops for retirement. Applicants respectfully submit that while Kahle does describe a *basic branch instruction* detection determination (see Kahle, col. 9 lines 56-58), Kahle does not teach nor suggest detecting *bogus branch* instructions. Kahle, does not teach nor suggest a method of recovering from and preventing basic branch instructions, let alone bogus branches. Kahle only teaches a method which depicts guest branch unit predicting the branch by reference to a conventional branch history table, (see Kahle, col. 10 lines 49-54), and further just indicating whether in this operation the predicted branch was taken. (See Kahle, col. 10 lines 54-56). Kahle does not teach flagging any other micro-ops *which pertain to the* at least one executed bogus branch micro-ops. Kahle only teaches storing the address of the *non-predicted* path in another path register, in order to permit recovery from misprediction. (See Khale, col. 10 lines 60-67). Storing the non-predicted, unrelated path (to permit recovery in case the incorrect branch instruction was

determined) is very different than flagging any other micro-ops which *pertain* to the at least one executed bogus branch micro-ops.

Furthermore, Examiner states that Kahle has not taught: decoding a first macro instruction into at least one micro-op; writing the at least one micro-op into a decoded micro-op cache. Applicants agree.

Examiner then states that it would be obvious to one of ordinary skill in the art to combine Kahle in view of McCrocklin, but see arguments infra.

Examiner states that Kahle and McCrocklin have not taught scrubbing a branch prediction logic storage buffer upon which the branch prediction logic is based. Applicants agree.

Examiner then states that it would be obvious to one of ordinary skill in the art to combine Kahle in view of Shiell and that Shiell teaches scrubbing a branch prediction logic storage buffer upon which the branch prediction logic is based. Applicants disagree. Shiell takes into account multiple predictions and branch history bits. (See Shiell, col. 14 lines 27-44). Shiell does not teach a method of scrubbing a branch prediction logic storage buffer upon which the branch prediction logic is based.

As per claim 14, Examiner states that Kahle has taught a method of preventing a bogus branch instruction from being executed in a microprocessor instruction pipeline, the method comprising: retiring the at least one instruction. Applicants respectfully submit that while Kahle does describe a *basic branch instruction* detection determination (see Kahle, col. 9 lines 56-58), Kahle, does not teach nor suggest a method of preventing a basic branch instruction, let alone bogus branches.

Examiner states that Kahle has not taught using microinstructions and writing at least one micro-op into a decoded micro-op cache. Applicants agree.

Examiner then states that it would be obvious to one of ordinary skill in the art to combine Kahle in view of McCrocklin, but see arguments infra.

Examiner states that Kahle and McCrocklin have not taught scrubbing a branch prediction logic storage buffer upon which the branch prediction logic is based. Applicants agree.

Examiner then states that it would be obvious to one of ordinary skill in the art to combine Kahle in view of Shiell and that Shiell teaches scrubbing a branch prediction logic storage buffer upon which the branch prediction logic is based. Applicants disagree. Shiell takes into account multiple predictions and branch history bits. (See Shiell, col. 14 lines 27-44). Shiell does not teach a method of scrubbing a branch prediction logic storage buffer.

As per claim 19, Examiner states that Kahle has taught an apparatus for detecting, recovering and preventing bogus branch instructions in a microprocessor, the method comprising: a branch prediction logic storage buffer for predicting whether a branch will be taken upon execution of the at least one decoded micro-op; an instruction execution unit for executing the at least one micro-op; and an instruction retirement unit which determines whether the at least one micro-op is of a bogus branch macro instruction, wherein if the instruction retirement unit determines the at least one micro-op is of a bogus branch macro instruction; any other micro-ops stored in the decoded micro-op cache pertaining to that bogus branch macro instruction are flagged and removed to the instruction retirement unit for retirement. Applicants respectfully submit that while Kahle does describe a *basic branch instruction* detection determination (see Kahle, col. 9 lines 56-58), Kahle does not teach nor suggest an apparatus for detecting *bogus* branch instructions. Furthermore, Kahle, does not teach nor suggest an apparatus for recovering

and preventing basic branch instructions, let alone bogus branch instructions. Kahle only teaches an apparatus which depicts guest branch unit predicting the branch by reference to a conventional branch history table, (see Kahle, col. 10 lines 49-54), and further just indicating whether in this operation the predicted branch was taken. (See Kahle, col. 10 lines 54-56). Kahle does not teach other micro-ops stored in the decoded micro-op cache pertaining to that bogus branch macro instructions are flagged. Kahle only teaches storing the address of the *non-predicted* path in other path register, in order to permit recovery from misprediction. (See Khale, col. 10 lines 60-67). Storing the non-predicted, unrelated path (to permit recovery in case the incorrect branch instruction was determined) is very different than flagging any other micro-ops stored in the decoded micro-op cache pertaining to that bogus branch macro instruction.

Examiner states that Kahle has not taught a decoded micro-op cache into which are written at least one decoded micro-op of a macro instruction. Applicants agree.

Examiner then states that it would be obvious to one of ordinary skill in the art to combine Kahle in view of McCrocklin, but see arguments infra.

Examiner states that Kahle and McCrocklin have not taught scrubbing a branch prediction logic storage buffer upon which the branch prediction logic is based. Applicants agree.

Examiner then states that it would be obvious to one of ordinary skill in the art to combine Kahle in view of Shiell and that Shiell teaches scrubbing a branch prediction logic storage buffer upon which the branch prediction logic is based. Applicants disagree. Shiell takes into account multiple predictions and branch history bits. (See Shiell, col. 14 lines 27-44). Shiell does not teach a method comprising, *inter alia*, the branch prediction logic storage buffer is scrubbed.

Claims 7-13 are rejected under 35 U.S.C. §103(a) as being unpatentable over Kahle et al., U.S. Patent No. 5,956,495 (hereinafter "Kahle") in view of McCrocklin et al, U.S. Patent No. 4,761,733 (hereinafter "McCrocklin").

As per claim 7, Examiner states that Kahle has taught a method of detecting bogus branch instructions in an instruction pipeline the method comprising: predicting whether a first instruction is a bogus branch instruction, looking ahead in the instruction pipeline to at least one second instruction related to the first instruction, wherein if the first operation is predicted to be a bogus branch, the method further comprises attaching a signal flag that indicates a bogus branch to the at least one second instruction. Applicants respectfully submit that while Kahle does describe a *basic branch instruction* detection determination (see Kahle, col. 9 lines 56-58), Kahle does not teach nor suggest detecting *bogus branch* instructions. Kahle does not teach looking ahead in the instruction pipeline to at least one second micro-op related to the first micro-op. Khale does not teach attaching a signal flag that indicates a *bogus* branch to the at least one second micro-op. Kahle only teaches storing the address of the *non-predicted* path in other path register, in order to permit recovery from misprediction. (See Khale, col. 10 lines 60-67). Storing the non-predicted, unrelated path (to permit recovery in case the incorrect branch instruction was determined) is very different than attaching a signal flag that indicates a bogus branch to the at least one second micro-op.

As per claim 10, Examiner states that Kahle has taught a method of recovering from a bogus branch instruction in a microprocessor instruction pipeline, the method comprising: determining whether a first instruction is a bogus branch, and deallocating from a decoded instruction cache at least one second instruction related to the first instruction. Applicants respectfully submit that while Kahle does describe a *basic branch*

instruction detection determination (see Kahle, col. 9 lines 56-58), Kahle does not teach nor suggest detecting *bogus branch* instructions. Furthermore, Kahle does not teach nor suggest a method of recovering from a basic branch instruction, let alone bogus branches. Kahle only teaches a method which depicts guest branch unit predicting the branch by reference to a conventional branch history table, (see Kahle, col. 10 lines 49-54), and further just indicating whether in this operation the predicted branch was taken. (See Kahle, col. 10 lines 54-56). Kahle does not teach deallocating at least one second instruction related to the first instruction, thereby recovering from a bogus branch instruction.

Furthermore, Examiner states that Kahle has not taught using microinstructions. Applicants agree.

Examiner then states that it would be obvious to one of ordinary skill in the art to combine Kahle in view of McCrocklin, but see arguments infra.

Claim Rejections under Combined References

As per claims 1, 7, 10, 14 and 19, Examiner then states that it would be obvious to one of ordinary skill in the art to combine Kahle in view of McCrocklin. However, Examiner does not point to a specific hint or suggestion to combine. Therefore, in addition and in the alternative to the individual claim arguments above, Applicants respectfully submit that there is no suggestion or motivation to combine Kahle and McCrocklin beyond the impermissible use of hindsight. Applicants submit that a *prima facie* case of obviousness has not been made. The MPEP requires that the references must suggest making the combinations. MPEP §2141.01 (citing Hodosh v. Block Drug

Co., Inc.); §706.02(j) (the initial burden is on the examiner to provide a convincing line of reasoning with explicit or implicit suggestions to combine references).

Merely stating that it would have been obvious for a person of ordinary skill in the art to combine references, without pointing to a specific hint or suggestion to combine, has been rejected by the Federal Circuit, as an invalid basis of rejection under 35 U.S.C. §103. See *In re Lee*, 277 F.3d 1338, 1343 (Fed. Cir. 2002). *See also, In re Dembiczak*, 175 F.3d 994,999 (Fed. Cir. 1999).

The Applicants therefore respectfully request Examiner to provide specific evidence of the teaching to combine Kahle and McCrocklin references or withdraw the rejection as improper.

Based on the foregoing arguments and amendments, Applicants respectfully submit that independent claims 1, 7, 10, 14 and 19 are allowable.

In addition, claims 2-6, 8-9, 11-13, 15-18 and 20-26 depend directly upon independent claims 1, 7, 10, 14 and 19, respectively. Therefore, Applicant respectfully submits that claims 2-6, 8-9, 11-13, 15-18 and 20-26 are allowable as being dependent on an allowable base claim.

In light of the arguments above, reconsideration and withdrawal of claims 1-26 under U.S.C. §103(a) is respectfully requested.

CONCLUSION

For all the above reasons, the Applicant respectfully submits that this application is in condition for allowance. A Notice of Allowance is earnestly solicited.

The Examiner is invited to contact the undersigned at (408) 975-7500 to discuss any matter concerning this application. The Office is hereby authorized to charge any additional fees or credit any overpayments under 37 C.F.R. § 1.16 or § 1.17 to Deposit Account No. **11-0600**.

Dated: January 15, 2004

By: Stephen T. Neal

Stephen T. Neal
(Reg. No. 47,815)
Attorneys for Intel Corporation

KENYON & KENYON
333 W. San Carlos Street
Suite 600
San Jose, CA 95110
Telephone: (408) 975-7500
Facsimile: (408) 975-7501